**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ**

**УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

ЛАБОРАТОРНАЯ РАБОТА №1

по дисциплине: **«Визуальные средства разработки программных**

**приложений»**

на тему: **Создание собственных классов и интерфейсов в *Java***

Выполнил: студент гр. ИТП-31

Дашкевич Д.А.

Принял: ассистент

Белявский Е. В.

Гомель 2019

**Цель**: изучить классы и интерфейсы в *Java*.

**Ход работы**

**Вариант 2**

# **Задание:**

Создать консольное приложение, удовлетворяющее следующим требованиям:

1. Создать собственные классы, интерфейсы.

2. Каждый класс должен иметь исчерпывающее смысл название и информативный состав.

3. При кодировании должны быть использованы соглашения об оформлении кода *java* *code* *convention*.

4. Классы должны быть грамотно разложены по пакетам.

5. Работа с консолью или консольное меню должно быть минимальным.

6. Для хранения параметров инициализации можно использовать файлы.

2. Новогодний подарок. Определить иерархию конфет и прочих сладостей. Создать несколько объектов-конфет. Собрать детский подарок с определением его веса. Провести сортировку конфет в подарке на основе одного из параметров. Найти конфету в подарке, соответствующую заданному диапазону содержания сахара.

**Результат работы программы:**

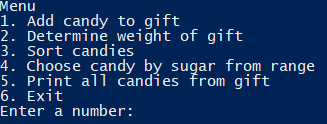
****

Рисунок 1 – Консольный вывод меню программы

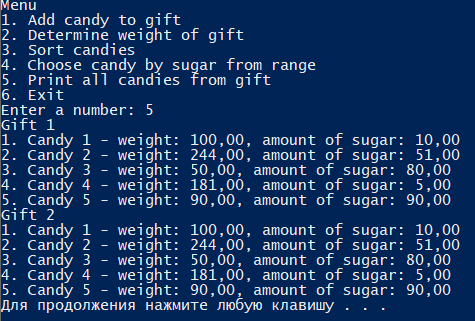


Рисунок 2 – Вывод в консоль значений параметров объектов класса Candy

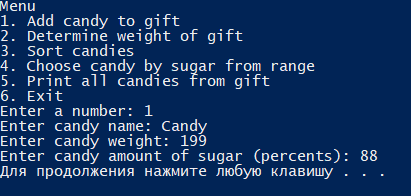


Рисунок 3 – Консольный ввод значений, инициализация объекта, добавление его в список

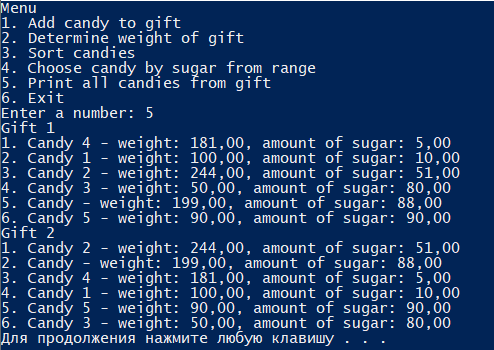


Рисунок 4 – Вывод в консоль значений параметров объектов класса Candy после использования метода сортировки объектов Gift

**Исходный код программы**

**Main.java:**

**public class** Main {  
  
 **private static** Scanner *in* = **new** Scanner(System.***in***);  
  
 **public static void** main(String[] args) {  
 List<Gift> gifts = **new** ArrayList<>();  
 gifts.add(**new** GiftImpl());  
 gifts.add(**new** BigGiftImpl());  
 *inputFromFile*(gifts);  
 **boolean** exit = **false**;  
 **while** (!exit) {  
 *printMenu*();  
 **int** k = *in*.nextInt();  
 **switch** (k) {  
 **case** 1:  
 *addCandy*(gifts);  
 *pause*();  
 **break**;  
 **case** 2:  
 **for** (Gift gift : gifts) {  
 **float** giftWeight = gift.getWeightOfCandies();  
 System.***out***.printf(**"Weight of gift: %.2f\n"**, giftWeight);  
 }  
 *pause*();  
 **break**;  
 **case** 3:  
 **for** (Gift gift : gifts) {  
 gift.sort();  
 }  
 System.***out***.println(**"Sort completed"**);  
 *pause*();  
 **break**;  
 **case** 4:  
 *chooseCandyBySugar*(gifts);  
 *pause*();  
 **break**;  
 **case** 5:  
 *printAllCandies*(gifts);  
 *pause*();  
 **break**;  
 **case** 6:  
 exit = **true**;  
 **break**;  
 }  
 }  
 }  
  
 **private static void** inputFromFile(List<Gift> gifts) {  
 System.***out***.print(**"Enter filename: "**);  
 String filename = *in*.next();  
 File file = **new** File(filename);  
 **if** (file.exists()) {  
 **try** {  
 BufferedReader reader = **new** BufferedReader(**new** FileReader(file));  
 String line = reader.readLine();  
 **while** (line != **null**) {  
 String[] parts = line.split(**", "**);  
 **float** weight = Float.*parseFloat*(parts[1]);  
 **float** amountOfSugar = Float.*parseFloat*(parts[2]);  
 Candy candy = **new** Candy(parts[0], weight, amountOfSugar);  
 **for** (Gift gift : gifts) {  
 gift.addCandy(candy);  
 }  
 line = reader.readLine();  
 }  
 } **catch** (IOException ex) {  
 ex.printStackTrace();  
 }  
 }  
 }  
  
 **private static void** printMenu() {  
 *clear*();  
 System.***out***.println(**"Menu"**);  
 System.***out***.println(**"1. Add candy to gift"**);  
 System.***out***.println(**"2. Determine weight of gift"**);  
 System.***out***.println(**"3. Sort candies"**);  
 System.***out***.println(**"4. Choose candy by sugar from range"**);  
 System.***out***.println(**"5. Print all candies from gift"**);  
 System.***out***.println(**"6. Exit"**);  
 System.***out***.print(**"Enter a number: "**);  
 }  
  
 **private static void** clear() {  
 **try** {  
 **new** ProcessBuilder(**"cmd"**, **"/c"**, **"cls"**).inheritIO().start().waitFor();  
 } **catch** (IOException | InterruptedException ex) {  
 System.***out***.println(ex.getMessage());  
 }  
 }  
  
 **private static void** pause() {  
 **try** {  
 **new** ProcessBuilder(**"cmd"**, **"/c"**, **"pause"**).inheritIO().start().waitFor();  
 } **catch** (IOException | InterruptedException ex) {  
 System.***out***.println(ex.getMessage());  
 }  
 }  
  
 **private static void** addCandy(List<Gift> gifts) {  
 System.***out***.print(**"Enter candy name: "**);  
 String name = *in*.next();  
 System.***out***.print(**"Enter candy weight: "**);  
 **float** weight = *in*.nextFloat();  
 System.***out***.print(**"Enter candy amount of sugar (percents): "**);  
 **float** amountOfSugar = *in*.nextFloat();  
 **try** {  
 Candy candy = **new** Candy(name, weight, amountOfSugar);  
 **for** (Gift gift : gifts) {  
 gift.addCandy(candy);  
 }  
 } **catch** (IllegalArgumentException ex) {  
 System.***out***.println(ex.getMessage());  
 System.***out***.println(**"Candy is not added"**);  
 }  
 }  
  
 **private static void** chooseCandyBySugar(List<Gift> gifts) {  
 System.***out***.print(**"Enter the minimum amount of sugar: "**);  
 **float** from = *in*.nextFloat();  
 System.***out***.print(**"Enter the maximum amount of sugar: "**);  
 **float** to = *in*.nextFloat();  
 Candy candy = gifts.get(0).chooseBySugarFromRange(from, to);  
 **if** (candy == **null**) {  
 System.***out***.println(**"Candy not found"**);  
 } **else** {  
 System.***out***.printf(**"Name of candy: %s, weight: %.2f, amount of sugar: %.2f\n"**, candy.getName(),  
 candy.getWeight(), candy.getAmountOfSugar());  
 }  
 }  
  
 **private static void** printAllCandies(List<Gift> gifts) {  
 **for** (**int** i = 0; i < gifts.size(); i++) {  
 List<Candy> listOfCandies = gifts.get(i).getCandies();  
 System.***out***.println(**"Gift "** + (i + 1));  
 **for** (**int** j = 0; j < listOfCandies.size(); j++) {  
 System.***out***.printf(**"%d. %s - weight: %.2f, amount of sugar: %.2f\n"**, j + 1, listOfCandies.get(j).getName(),  
 listOfCandies.get(j).getWeight(), listOfCandies.get(j).getAmountOfSugar());  
 }  
 }  
  
 }  
}

**Gift.java:**

**public interface** Gift {  
 **void** addCandy(Candy candy);  
 ArrayList<Candy> getCandies();  
 **float** getWeightOfCandies();  
 **void** sortBySugar();  
 Candy chooseBySugarFromRange(**float** from, **float** to);  
}

**Candy.java:**

**class** Candy {  
 **private** String **name**;  
 **private float weight**;  
 **private float amountOfSugar**;  
  
 Candy(String name, **float** weight, **float** amountOfSugar) **throws** IllegalArgumentException {  
 **this**.**name** = name;  
 **if** (weight <= 0f) {  
 **throw new** IllegalArgumentException(**"Weight mustn't be negative or zero"**);  
 }  
 **this**.**weight** = weight;  
 **if** (amountOfSugar < 0f || amountOfSugar > 100f) {  
 **throw new** IllegalArgumentException(**"Sugar must be in range(0, 100)"**);  
 }  
 **this**.**amountOfSugar** = amountOfSugar;  
 }  
  
 String getName() {  
 **return name**;  
 }  
  
 **float** getWeight() {  
 **return weight**;  
 }  
  
 **float** getAmountOfSugar() {  
 **return amountOfSugar**;  
 }  
}

**GiftImpl.java:**

**class** GiftImpl **implements** Gift {  
  
 **private** List<Candy> **candies** = **new** ArrayList<>();  
  
 @Override  
 **public void** addCandy(Candy candy) {  
 **candies**.add(candy);  
 }  
  
 @Override  
 **public** List<Candy> getCandies() {  
 **return candies**;  
 }  
  
 @Override  
 **public float** getWeightOfCandies() {  
 **float** weight = 0;  
 **for** (Candy c : **candies**) {  
 weight += c.getWeight();  
 }  
 **return** weight;  
 }  
  
 @Override  
 **public void** sort() {  
 **candies**.sort((c1, c2) -> (**int**) (c1.getAmountOfSugar() - c2.getAmountOfSugar()));  
 }  
  
 @Override  
 **public** Candy chooseBySugarFromRange(**float** from, **float** to) {  
 **for** (Candy c : **candies**) {  
 **if** (c.getAmountOfSugar() >= from && c.getAmountOfSugar() <= to) {  
 **return** c;  
 }  
 }  
 **return null**;  
 }  
}

**BigGiftImpl.java:**

**public class** BigGiftImpl **implements** Gift {  
  
 **private** List<Candy> **candies** = **new** ArrayList<>();  
  
 @Override  
 **public void** addCandy(Candy candy) {  
 **candies**.add(candy);  
 }  
  
 @Override  
 **public** List<Candy> getCandies() {  
 **return candies**;  
 }  
  
 @Override  
 **public float** getWeightOfCandies() {  
 **float** weight = 0;  
 **for** (Candy c : **candies**) {  
 weight += c.getWeight();  
 }  
 **return** weight;  
 }  
  
 @Override  
 **public void** sort() {  
 **candies**.sort((c1, c2) -> (**int**) (c2.getWeight() - c1.getWeight()));  
 }  
  
 @Override  
 **public** Candy chooseBySugarFromRange(**float** from, **float** to) {  
 **for** (Candy c : **candies**) {  
 **if** (c.getAmountOfSugar() >= from && c.getAmountOfSugar() <= to) {  
 **return** c;  
 }  
 }  
 **return null**;  
 }  
}

**Вывод:** в результате выполнения лабораторной работы были изучены классы и интерфейсы в *Java* и написана программа, которая использует иерархию классов, консольный ввод-вывод, работу с файлами.